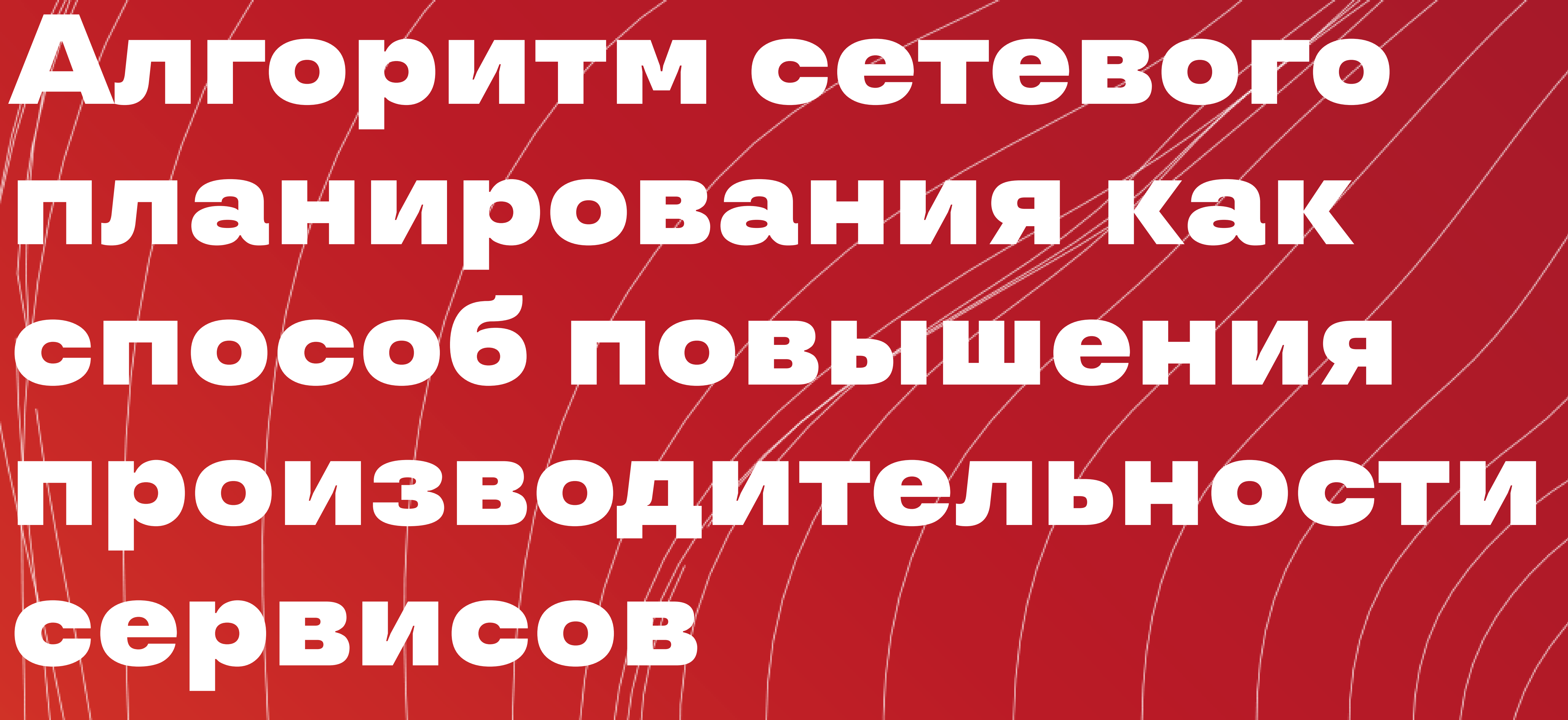


Алгоритм сетевого планирования как способ повышения производительности сервисов

Игорь Чирков (Почтатех)

HL++ HighLoad++ 2022

почтатех

Разрабатываем сложные цифровые продукты Почты России



pochta.tech

120
ИТ-продутов для Почты России

1500
сотрудников

>16 000 000
ежемесячная аудитория приложения и портала

- Приложения iOS, Android
- Сервис мониторинга транспорта Почты России
- Сайт pochta.ru (ПОЧТА РОССИИ)
- Почтовые сервисы для бизнеса
- Интерфейсы для курьеров, системы для почтальонов
- Прием и отправка заказных электронных писем



Москва, Сколково, Петербург, Пушкин, Омск, Ижевск, Иннополис

О проекте Личный кабинет юридического лица



otpravka.pochta.ru

2015

начало
разработки

2021 **206 187 575**

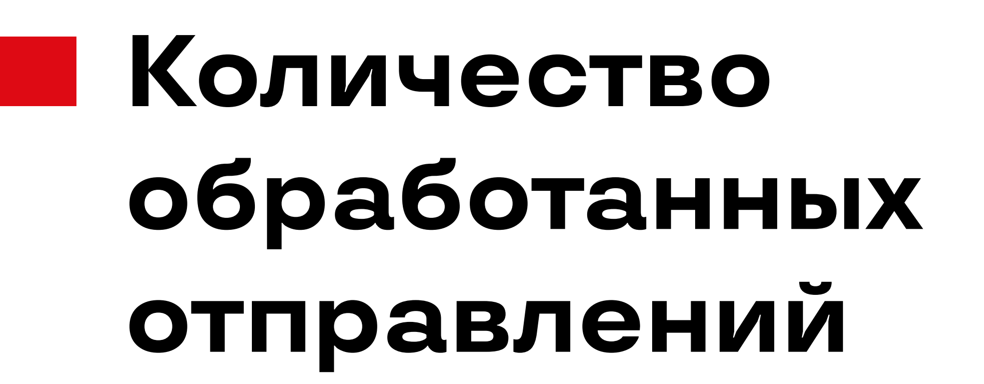
2022 **>220 000 000**

ежегодный рост количества отправлений
через наш портал

Наш проект демонстрирует высокую динамику развития как в плане выручки, так и в росте нагрузки



Динамика роста нагрузки начиная с 2016 года



миллионов

250
200
150
100
50
0

Год	2016	2017	2018	2019	2020	2021	2022
Значение	0,12	3,03	28,97	103,37	148,63	205,86	220



Структура REST-сервиса

Controller

Service

External

Systems

DB



Структура сервисного слоя после первой фазы рефакторинга

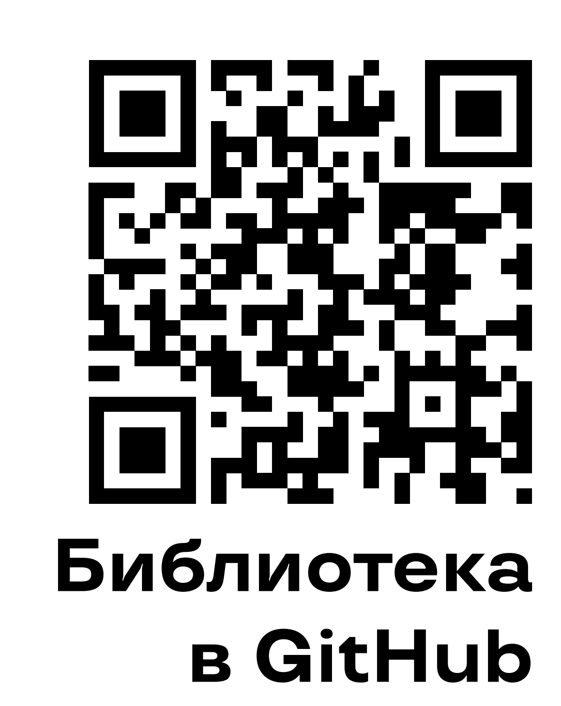
A0 —a→ A1 —b→ A2 —c→ A3 —d→ A4 —e→

A5 —f→ A6 —g→ A7 —h→ A8 —i→ A9

Возможны также логические ветвления



Оценка среднего времени выполнения операций



A0 —a, 2→ A1 —b, 3→ A2 —c, 3→ A3 —d, 4→ A4 —e, 1→

A5 —f, 2→ A6 —g, 5→ A7 —h, 1→ A8 —i, 1→ A9

Общее время составляет 22 единицы



Постановка задачи

Ускорить работу сервиса, организованного как последовательное выполнение нескольких затратных по времени операций, зависящих друг от друга в плане данных.



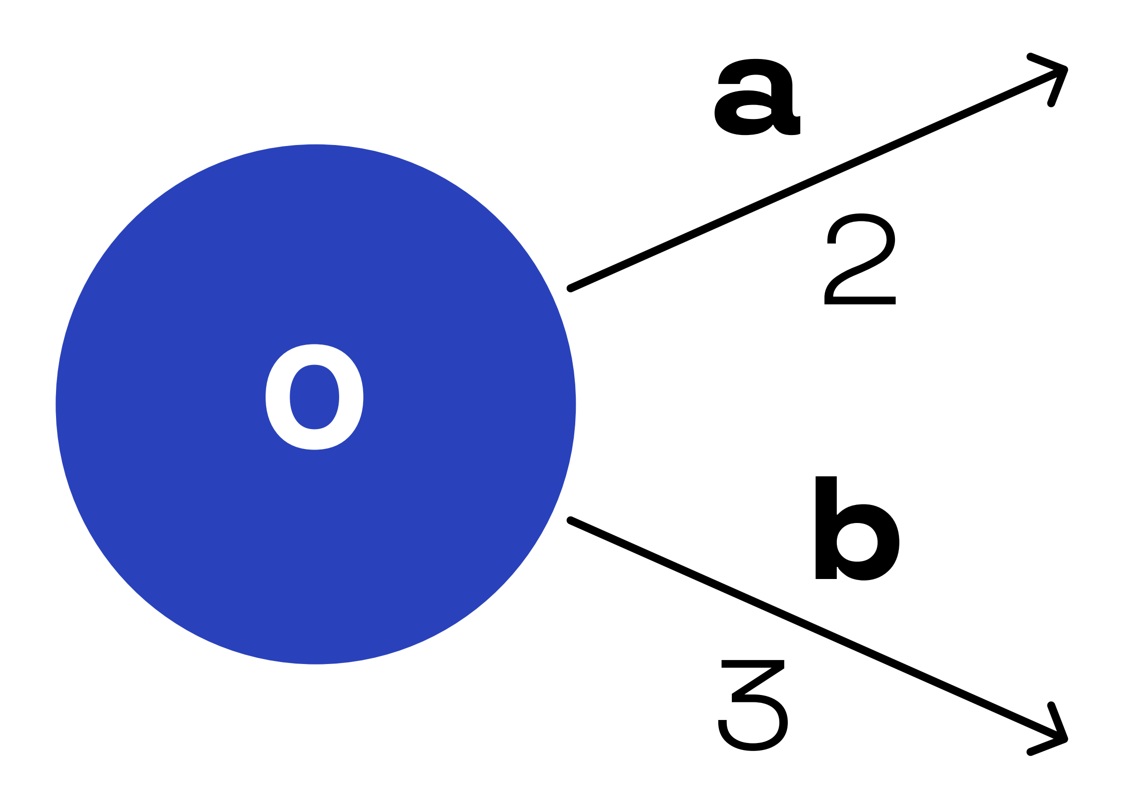
Пути решения

- Оптимизировать выполнение отдельных операций
- Изменить структуру сервиса,
  перейти на многопоточную обработку
- Совместить первый и второй подходы
  (решить задачу сетевого планирования)



Пример построения сети

a, b – зависят только от входящих параметров





Пример построения сети

c – зависит от результатов a
d, e – от результатов b

1
a
2
c
3
0
b
3
d
4
2
e
1



Пример построения сети

f, g – от c, d

0 → 1: a, 2
0 → 2: b, 3
1 → 3: c, 3
2 → 3: d, 4
2 →: e, 1
3 →: f, 2
3 →: g, 5



Пример построения сети

h – от f


0 1 2 3 4

a 2
b 3
c 3
d 4
e 1
f 2
g 5
h 1



Пример построения сети

i – от e, g



Разделение сети на слои

Так выглядит наш граф после удаления исходящих из истока ребер

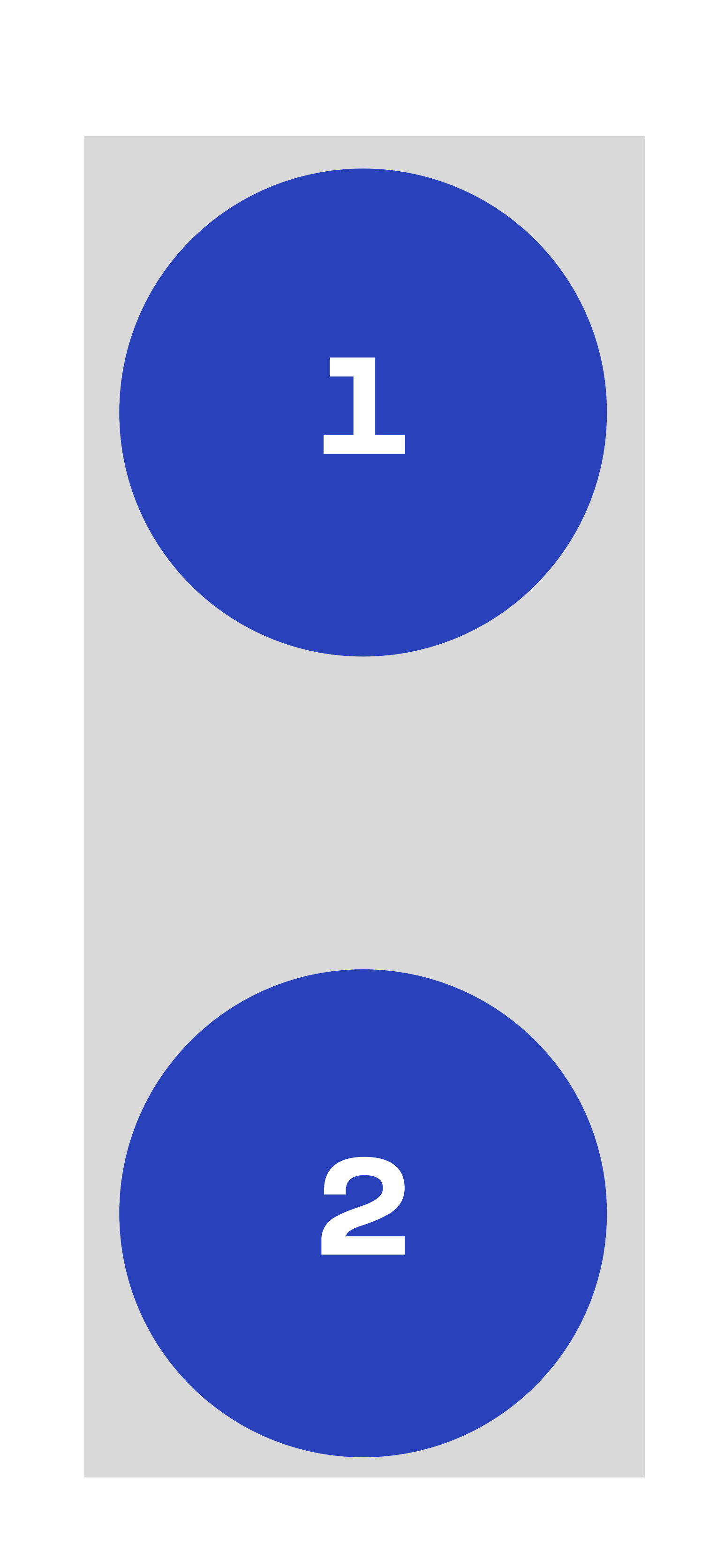


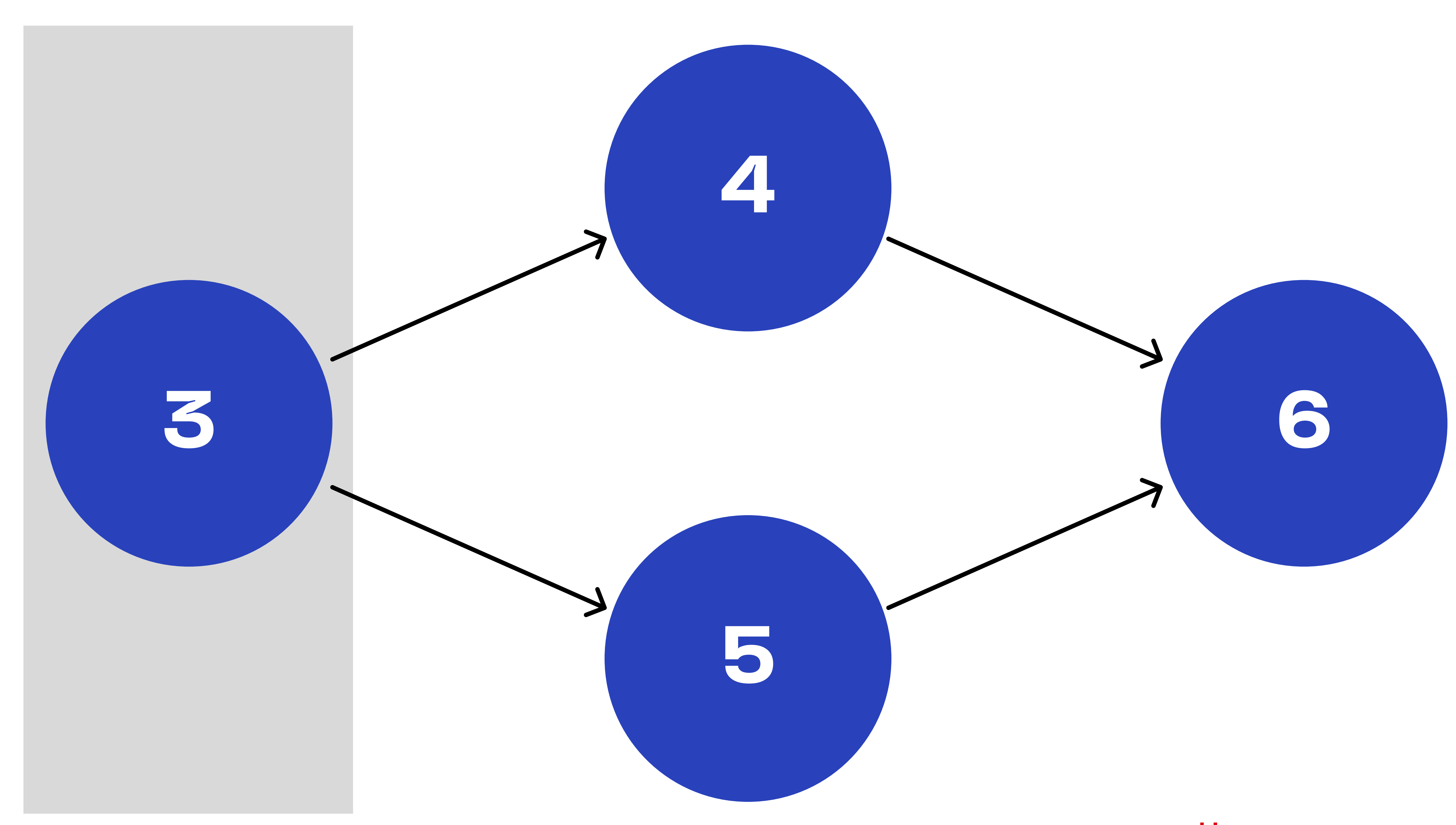


Разделение сети на слои

Вторая итерация






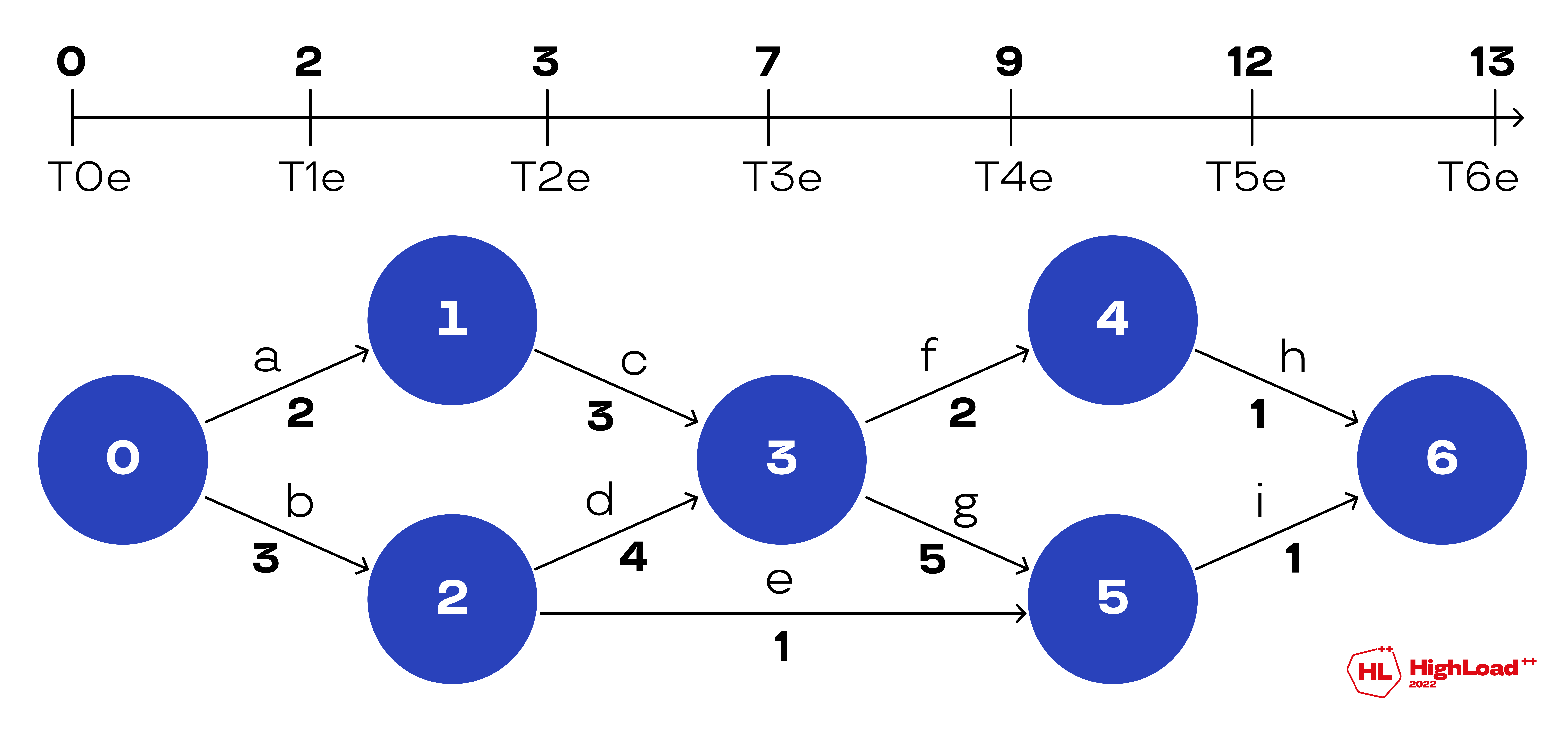


Разделение сети на слои

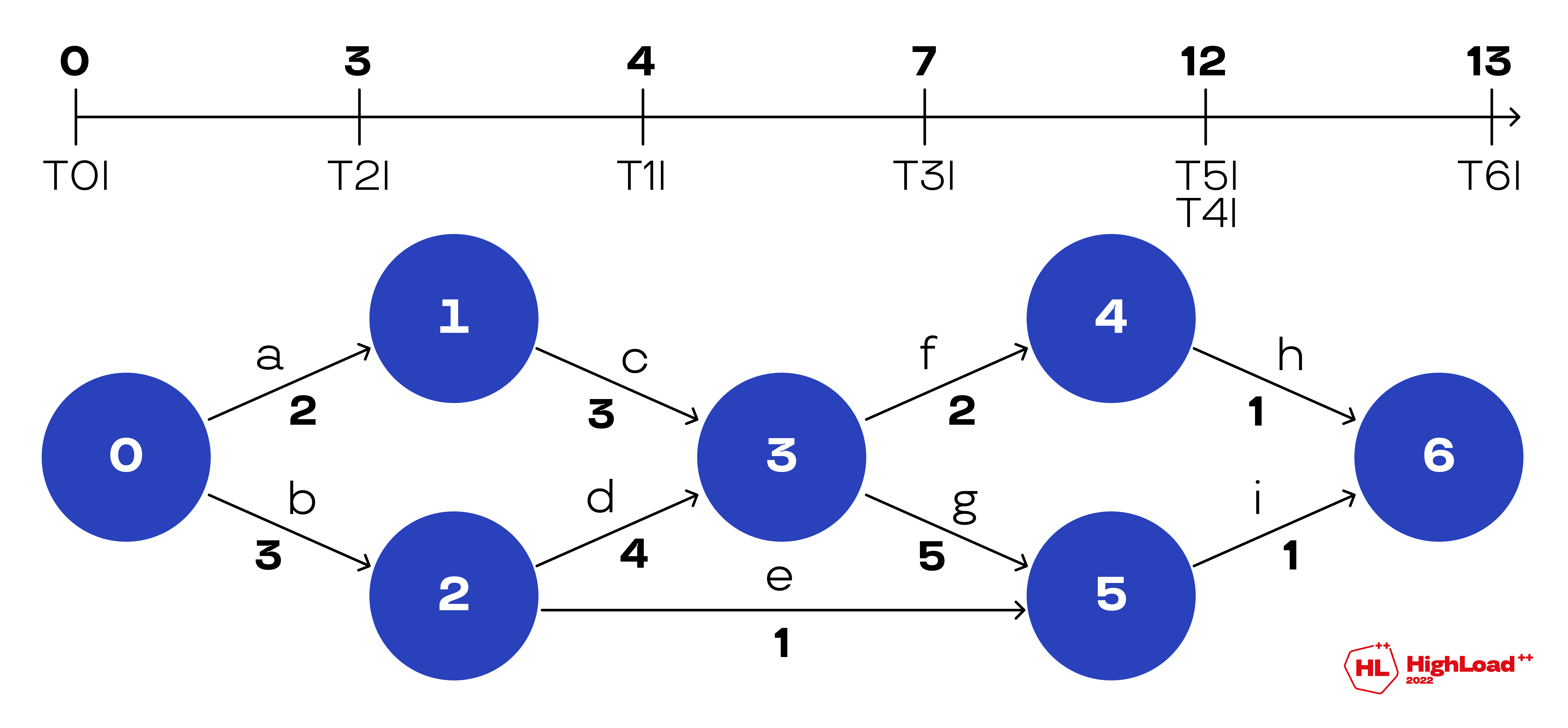
Финальный результат



Вычисление ранних времен



Вычисление поздних времен




Результаты вычислений

T0e = 0	T0l = 0
T1e = 2	T1l = 4
T2e = 3	T2l = 3
T3e = 7	T3l = 7
T4e = 9	T4l = 12
T5e = 12	T5l = 12
T6e = 13	T6l = 13



Нахождение критического пути

Выделенные стрелки соединяют вершины с совпадающими ранними и поздними временами



Подведение итогов

- Улучшилась читаемость кода
- Получили выигрыш во времени за счет распараллеливания операций
- Нашли критические операции, которые стоит оптимизировать



Результаты для нашего проекта

- Сервис, обрабатывающий запрос, включает 13 операций
- Критический путь является единственным и содержит 6 операций
- Среднее время обработки запроса снизилось более чем в 2 раза





Оставить отзыв о докладе

Igor.Chirkov@russianpost.ru